Verification of Compositional Software Architectures

by Diana Lee and Bob Filman

November 4, 1997

Microelectronics and Computer Technology Corporation's (MCC) Object Infrastructure Project (OIP) is developing a reference architecture to automate the implementation of system-wide requirements in a system assembled from components [1]. The premise of that work is that good qualities can accrue to a system by manipulating its inter-component communications [2]. As part of verifying this claim, we have developed a *framework* (the Object Infrastructure Framework, or OIF) to manipulate inter-object communication and a testbed application. This paper describes that test application, some elements of the framework, and the empirical results of experiments on our thesis.

To be useful as an evaluation of the reference architecture, the test application should be:

- **Quantitative**: The test cases should yield quantitative results that can be used to measure success.
- **Flexible**: The test application should lend itself to variation for exploring different elements of the problem space.
- **Realistic**: The test application should have applicability to the "real world" and model issues in the customers' domain.
- **Understandable**: The test application should be easy to grasp. It should not require a deep understanding of a particular application domain.
- **Demonstrable**: Research requires demonstration, and a test application that can double as a demonstration system saves much effort.

The sponsors of this work are (by and large) organizations concerned with building large ill-structured distributed systems (in contrast to the more structured systems of, say, transactional information systems.) The systems are characterized by dynamic faults, the importance of hard and soft real-time responses, and open environments. Examples of such systems include communication architectures and battlefield systems.

In response to these needs, the Object Infrastructure Project has developed a test application, *Vendoom*. This paper discusses the features of Vendoom that have made it a valuable exploration and evaluation tool for a component-based, distributed, real-time computing environment, and provides an example of how Vendoom is being used to evaluate the Object Infrastructure reference architecture.

Vendoom Overview

The first goal of Vendoom is to create an abstract application that is easily understood and is able to model key business drivers in the real world. Business drivers include issues of:

- **Quality of service**: Many tasks compete for limited computational resources. It is important to give better access to higher priority tasks. Examples of quality of service issues are calls competing for resources in a communications network (where, 911 calls are more important than ordinary service, and there is likely more demand for satellite channels than capacity) or in a battle management system, where resources such as AWACs or data analysis may have to be distributed to sub-battles. (There is also the

issue of really reserving underlying communication bandwidth a la ATM or IPv6, but hardware or network dependencies are deemed beyond the scope of this work.)

- **Failure diagnosis and repair**: In systems composed of communicating distributed heterogeneous components, detecting failures and assigning blame become more complex. This is particularly important for systems that cross administrative domains (such as multi-vendor telephone connections) and for programmers actually trying to debug concurrent systems.
- **Dynamic component invocation**: Many business tasks are complex requiring more than one step of computational processing. An example is an AWAC battle management scenario in which an AWAC image is routed through a network of satellites and a network of ground stations. In addition to the relay functions, nodes in this network may also perform some functions such as image processing.

The goal is to capture these business drivers in a test application environment without descending to the details of particular applications. We want the test application to remain general and flexible enough to abstractly model many different industries.

In response to these requirements, we created the "game" Vendoom. The original inspiration was the idea of "distributed Doom" (a shoot-em-up game). A large part of the charm of such games rests on the clever animations, and seeking to avoid the major time-sink of clever graphics, we recalled the original computer role game, Don Wood's Adventure, which provided a purely textual tour through a maze of twisty passages. Combining the two and inflating to a high degree of concurrency gets us Vendoom. Vendoom posits *teams* of *players* that wander through a maze of *rooms*, finding *treasures* (some of which are poisonous) and shooting each other. It's easy in that scenario to assign points for treasures and good shots, and to therefore rate algorithms on the relative score they achieve.

Well that fine, but we need coordination, interaction at beyond the "you're dead" level, and contention for resources. The element that elevates Vendoom from being a game with bad graphics to a test application that addresses business drivers is a network of controllers and sensors. Players, while in the maze of rooms, have limited vision. They can see other players and other objects, but they cannot determine other players' teams or treasure's values. Each room has sensors that can provide that information, but to obtain it the players must "call" the appropriate sensor. The call is routed through a network on controllers. The contest between two players of different teams thus becomes the question of whose network will route the call to the sensor first (enabling that player to kill before being killed.) By varying the player and controller algorithms, different elements of the distributed coordination space can be explored. This relationship is illustrated in Figure 1.

Figure 1...

The quality of service business driver is modeled by overloading the network of controllers and sensors. By doing so, it is possible to have a number of tasks waiting on the limited controller resource. We can test quality of service by varying the point value of different players. Since it's better not to have higher-valued players shot, a clever network will give their requests higher priority. The "framework" question is whether this priority-based scheduling can be achieved by the framework, independent of the network controller implementation code.

The communications network business driver is modeled by the network connection issues of sensors and controllers. (Enough that one characterization of Vendoom is "competitive cell phoning.") The network of controllers can also be understood as successive processing functions of a multi-step processing problem.

By dynamically making controllers fail, we can test the ability of controller network algorithms to respond to failure. And the mere debugging of systems with a few hundred concurrent tasks has given us considerable experience in useful framework support for debugging. By including network control algorithms with routing information, data replication

can become important. And by allowing networks to spy on each other (and inject alternative messages) issues of security can be addressed.

A Vendoom Scenario

A Vendoom scenario might play as follows: Sir Lancelot of the blue team enters the Throne Room and sees Player A, Player B, and an object T. He also sees the blue team's sensor, number 555-1234. Sir Lancelot calls his controller asking to be connected to 555-1234. Sir Lancelot's controller routes the request to sensor 555-1234 which replies: in the Throne Room there is Sir Lancelot of the blue team worth –10 points, King Arthur of the blue team worth –100 points, The Wicked Mordred of the red team worth 20 points, the Holy Grail worth 15 points, and exits to the north, south, and east. (In the meantime, King Arthur and The Wicked Mordred are simultaneously calling their sensors to get information). Sir Lancelot shoots The Wicked Mordred (earning 20 points for the blue team) and picks up the Holy Grail (earning another 15 points).

Vendoom: Quantitative and Flexible

Vendoom produces quantitative data useful for evaluation. Because the ability of the team to win is dependent on their ability to get messages through their network, scores becomes a quantitative measure of effective use of the network. This feature makes Vendoom valuable in evaluating distribution and network issues.

Vendoom is flexible enough to adapt to evolving and changing business drivers. This flexibility comes with the ability to change the rules of the game and the ability to change the behaviors of the players. This feature of Vendoom was used early as the project honed the telecommunications business drivers. For example, it turns out that an effective strategy in Vendoom can be to be a Rambo: on entering a room, shoot first and call later. This strategy can be countered in the game design by introducing (a team of) civilians, inanimate players that are nevertheless expensive to kill. A more interesting variation became clear in the first large-scale competitions: teams that were behind caught up. It turns out that as teams lost players, the congestion on that team's controller network declined, enabling better response for the remaining players. This artifact was especially inappropriate as a model for telecommunications, where failure to complete a call is likely to provoke another attempt, not permanent destruction of the cell phone. We extended the rules to allow a version where shot players were merely rendered ineffective while they remained in the same room, a trivial but successful extension. These points illustrate the flexibility of the underlying model, and the guiding modeling principle that if the current rules do not capture the business model, change the rules.

Vendoom can be used to evaluate network path algorithms, network failure detection and recovery algorithms, queue triage mechanisms that purges tasks in the queue whose due date has expired, queuing algorithms, and central planning algorithms that coordinate the activities of all the players and sensors on a team.

The Object Infrastructure Project's Reference Architecture

MCC's Object Infrastructure Project (OIP) is using Vendoom to evaluate its architecture for component-based applications. OIP is addressing the problem of imbuing a system assembled from software components with system-wide properties such as security, reliability, consistency, manage-ability, and real-time quality of service, an other such "ilities" [1]. Because the scope of these properties extends beyond a single component and cannot be encapsulated, it is difficult to implement these "ilities" without interleaving and tangling "ility" code within the components and thereby destroying the "component-ness" of the system.

To address this difficulty, the Object Infrastructure is developing a reference architecture (The Object Infrastructure Framework, or OIF) that enables "ilities" with minimal disruption to the components.

Quality of Service

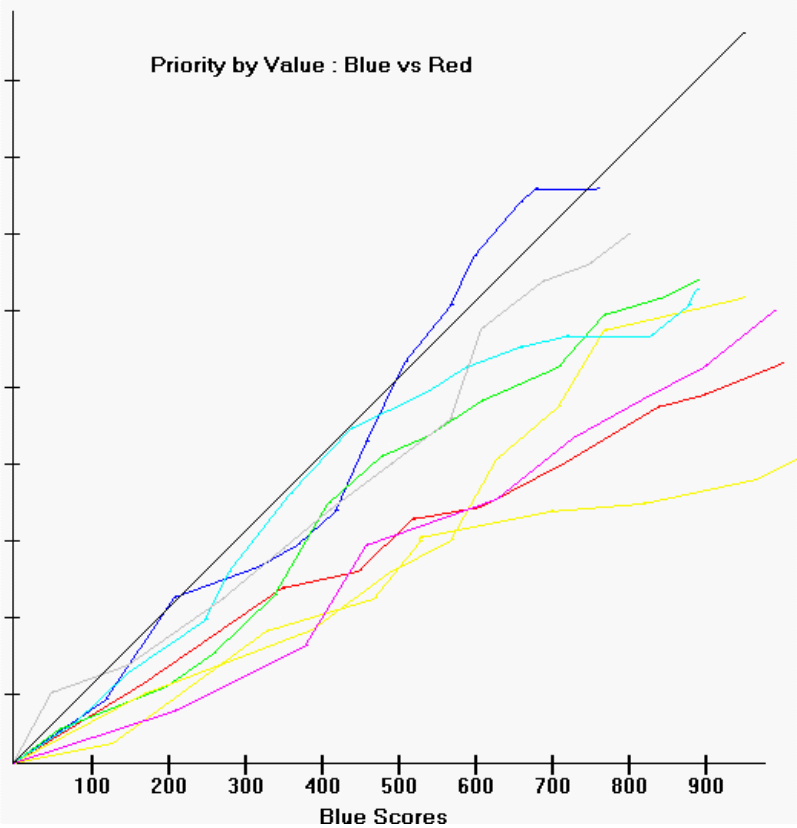
Our first experiment was to develop a version of OIF, implement Vendoom on that version and demonstrate that the framework itself could be used to control quality of service to the overriding application. Here we use Quality of Service as synonymous with "soft real time." Soft real time systems are characterized by "higher priority" tasks being given preferential allocation of computational resources. The Object Infrastructure Framework implements this preferential treatment in a queue control mechanism that selects, from the available tasks waiting for processor time, the highest priority task to execute next.

Using Vendoom to Explore Quality of Service Issues

Inserting quality of service behavior into Vendoom required only two small (one line) changes to the application.

1. The player constructor was modified to set the default message priority equal to the player's value, and
2. The controller constructor was modified to ask to receive messages in priority order.

We ran several test cases with the blue team modified by these changes (while the egalitarian red team continued to process requests first-in-first-out.) Figure 2 illustrates the results. In the figure, the dark gray line, a 45-degree line, represents the points in time that the blue team score is equal to the red team score. Points below the 45 degree lines represent points that the blue team score is



greater than the red team score. Points above the 45-degree line represent points that the red team score is greater than the blue team score. The figure shows in these 8 trials, the vast majority of data points is below the 45 degree line where the blue team (the one with priority based queue control) is winning.

From the figure, one concludes the Object Infrastructure Framework is successful at impacting the quality of service. The blue team that implemented the priority queue quality of

service gained a definite advantage by allowing their more valuable players better access to the network of controllers. What is significant about these trials is that the Object Infrastructure Framework, with minimal impact or disturbance to the Vendoom components, achieves quality of service in Vendoom. Quality of service is achieved and the components remain independent and reusable. Our next planned experiments will contrast scheduling by due date versus priority.

Summary

We have described a test-bed scenario, Vendoom, that has proven useful in evaluating a framework for constructing component-based systems. Vendoom gives quantitative data for measuring the effectiveness of frameworks and algorithms; its scenario captures key business drivers for a variety of domains. Vendoom has also provided a flexible environment, suggesting extensions for dealing with issues such as fault diagnosis and repair, load balancing, security and reliability.. Lastly, the rules of Vendoom game are easy to understand, and the game and quantitative results make for good demonstrations of a project's achievements.

Acknowledgements

References

[1] Ted Linden, "Object Infrastructure for Developing Large, Distributed Applications," OOPSLA 97, Panel on Exploring Largeness, Complexity & Scalability from the OOT perspective, Atlanta, Oct. 1997, http://www.mcc.com/projects/oip/dist_app.html

[2] Robert E. Filman, "Achieving Ilities," submitted to OMG-DARPA-MCC Workshop On Compositional Software Architectures, January 1998.